**INHERITANCE IN HIBERNATE**

**Inheritance through mapping files(XML files)**

Same as in Java, hibernate also have some inheritance concepts which can be achieved through Hibernate Mapping Files.

Inheritance in hibernate is achieved through 3 ways they are :

1. Subclass tag : - in this we will create a discriminator tag which can be used as a sub class referee.

Syntax for discriminator tag :

<discriminator column=*"column name"* type=*"column type"* length=*"20"*></discriminator>

Mapping class for subclass tag:

<hibernate-mapping>

<class name=*"com.motivity.Payment"* table=*"payment"*>

<id name=*"payment\_id"* column=*"pid"*></id>

<discriminator column=*"paytype"* type=*"string"* length=*"20"*></discriminator>

<property name=*"amount"* column=*"amount"*></property>

<subclass name=*"com.motivity.CreditCard"* discriminator-value=*"CC"*>

<property name=*"cctype"* column=*"cctype"*></property>

</subclass>

<subclass name=*"com.motivity.Cheque"* discriminator-value=*"CQ"*>

<property name=*"chequeType"* column=*"chequetype"*></property>

</subclass>

</class>

</hibernate-mapping>

Configuration will be same as all the hibernate files have.

When we use subclass tag we only get to create one database table and based on the discriminator value the column values will be loaded.

2.Joined-subclass tag : joined subclass will create different tables for each of the parent and the child class.

This will use key as a foreign key constraint from the parent class.

Mapping class for joined-subclass tag:

<hibernate-mapping>

<class name=*"com.motivity1.Payment"* table=*"payment"*>

<id name=*"payment\_id"* column=*"pid"*></id>

<property name=*"amount"* column=*"amount"*></property>

<joined-subclass name=*"com.motivity1.Cheque"* table=*"cheque"*>

<key column=*"cid"*></key>

<property name=*"chequeType"* column=*"chequetype"*></property>

</joined-subclass>

<joined-subclass name=*"com.motivity1.CreditCard"* table=*"creditcard"*>

<key column=*"ccid"*></key>

<property name=*"cctype"* column=*"cctype"*></property>

</joined-subclass>

</class>

</hibernate-mapping>

1. Union-subclass :- union subclass creates tables only for the child class

Mapping class for union-subclass tag:

<hibernate-mapping>

<class name=*"com.motivity2.Payment"* table=*"payment"*>

<id name=*"payment\_id"* column=*"pid"*></id>

<property name=*"amount"* column=*"amount"*></property>

<union-subclass name=*"com.motivity2.Cheque"* >

<property name=*"chequeType"* column=*"chequetype"*></property>

</union-subclass>

<union-subclass name=*"com.motivity2.CreditCard"* >

<property name=*"cctype"* column=*"cctype"*></property>

</union-subclass>

</class>

</hibernate-mapping>

Here, we can also use multiple resource files in a single configuration file but there are some issues with it as when we load the main class it will consider the first mapping file.

<hibernate-configuration>

<session-factory>

<property name=*"connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/lotus</property>

<property name=*"connection.user"*>root</property>

<property name=*"connection.password"*>root</property>

<property name=*"show\_sql"*>true</property>

<property name=*"dialet"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hbm2ddl.auto"*>create</property>

<mapping resource=*"Payment1.xml"*></mapping>

<mapping resource =*"Payment2.xml"*></mapping>

<mapping resource =*"payment3.xml"*></mapping>

</session-factory>

</hibernate-configuration>

Here the issue is that the mapping resource payment1 will be selected each time we load the main class

To rectify it we can comment the resources which we don’t want to load and we can load the resource we wanted to load.

**Hibernate Inheritance without using xml files(using Annotations):**

Using annotations is one of the best ways to implement the inheritance in hibernate.

However the configuration file will be same and we only need to change the mapping resource to mapping class since we create the annotations in the java class directly.

Below is the example for the configuration file for annotations.

<hibernate-configuration>

<session-factory>

<property name=*"connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/lotus</property>

<property name=*"connection.user"*>root</property>

<property name=*"connection.password"*>root</property>

<property name=*"show\_sql"*>true</property>

<property name=*"dialet"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hbm2ddl.auto"*>create</property>

<mapping class=*"com.annotations.Payment"*/>

<mapping class=*"com.annotations.CreditCard"*/>

<mapping class=*"com.annotations.Cheque"*/>

</session-factory>

</hibernate-configuration>

**Details of annotations :**

**For single table :**

Payment 15 is the main table

@Entity

@Table(name="payment15")

@Inheritance(strategy=InheritanceType.***SINGLE\_TABLE***)

@DiscriminatorColumn(name="paymenttype",discriminatorType=DiscriminatorType.***STRING***,length=20)

**public** **class** Payment {

@Id

@Column(name="pid")

**private** **int** pid;

@Column(name="amount")

**private** **int** amount;

//we also create getters and setters for the variables we create

}

This is for the one of the child class

@Entity

@DiscriminatorValue("CQ")

**public** **class** Cheque **extends** Payment {

@Column(name="ctype")

**private** String chequetype;

//generate setters and getters

}

This is for another child class

@Entity

@DiscriminatorValue("CC")

**public** **class** CreditCard **extends** Payment{

@Column(name="cctype")

**private** String cctype;

//generate setters and getters

}

Here the discriminator will be the one who judge to load the values in columns and where the values are to be inserted.

@Inheritance(strategy=InheritanceType.***SINGLE\_TABLE***)

The above line depicts that the inheritance type we used is single table which means only one table will be created and all the columns from the child classes will be loaded into this table only.

**For joined type :**

In this scenario we will be able to create 2 tables and they will be only for the child classes but not for the parent class.

Parent class :

@Entity

@Table(name="payment")

@Inheritance(strategy=InheritanceType.***JOINED***)

**public** **class** Payment {

@Id

@Column(name="pid")

**private** **int** pid;

@Column(name="amount")

**private** **int** amount;

//setters and getters

}

Child class 1:

@Entity

@Table(name="cheque")

@PrimaryKeyJoinColumn(name="pid")

**public** **class** Cheque **extends** Payment {

@Column(name="chequetype")

**private** String chequetype;

//setters and getters

}

Child class 2 :

@Entity

@Table(name="creditcard")

@PrimaryKeyJoinColumn(name="pid")

**public** **class** CreditCard **extends** Payment{

@Column(name="cctype")

**private** String cctype;

//setters and getters

}

@Inheritance(strategy=InheritanceType.***JOINED***)

The above line depicts that the type joined means all the columns in the parent class will be joined to the table of child class.

The tables will be created based on the values we are loading in the main class.

3.Table per class

Table per class means that for every class we will create a table and we will use foreign key

Parent class:

@Entity

@Table(name="payment1")

@Inheritance(strategy=InheritanceType.***TABLE\_PER\_CLASS***)

**public** **class** Payment {

@Id

@Column(name="pid")

**private** **int** pid;

@Column(name="amount")

**private** **int** amount;

//generate setters and getters

}

Child class 1:

@Entity

@Table(name="cheque1")

@PrimaryKeyJoinColumn(name="pid")

**public** **class** Cheque **extends** Payment {

@Column(name="chequetype")

**private** String chequetype;

//setters and getters

}

Child class 2:

@Entity

@Table(name="creditcard1")

@PrimaryKeyJoinColumn(name="pid")

**public** **class** CreditCard **extends** Payment{

@Column(name="cctype")

**private** String cctype;

//setters and getters

}

In this we will be able to create 3 tables one for each class and we will be taking foreign key from the parent class